

# Programmation Android

## 10 - Spatialite

Yann Caron

ENSG  
Géomatique

# Sommaire - Spatialite

- Présentation
- Installation
- Modélisation de base de donnée
- Création de base de donnée
- Spatialite Helper
- Requêtes
- Objets géométriques
- Pour aller plus loin !



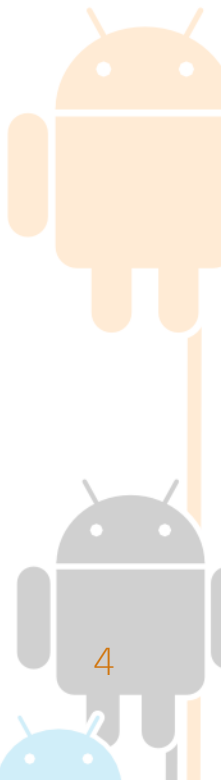
# IN01 – Séance 10

## Présentation

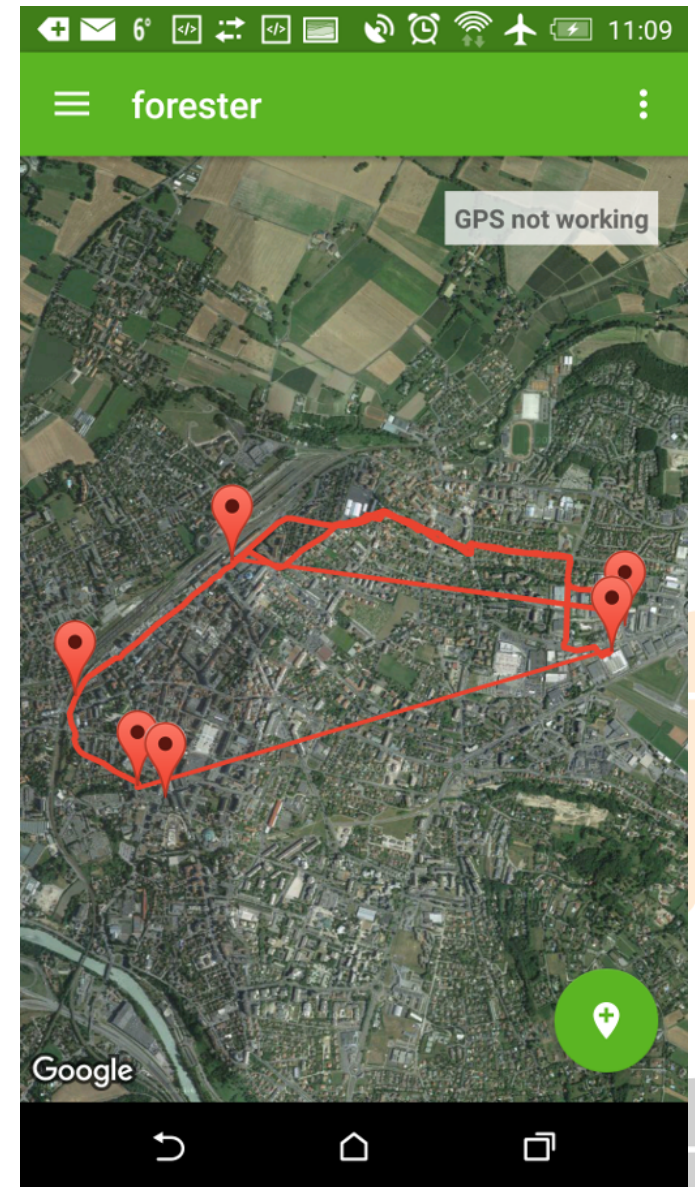
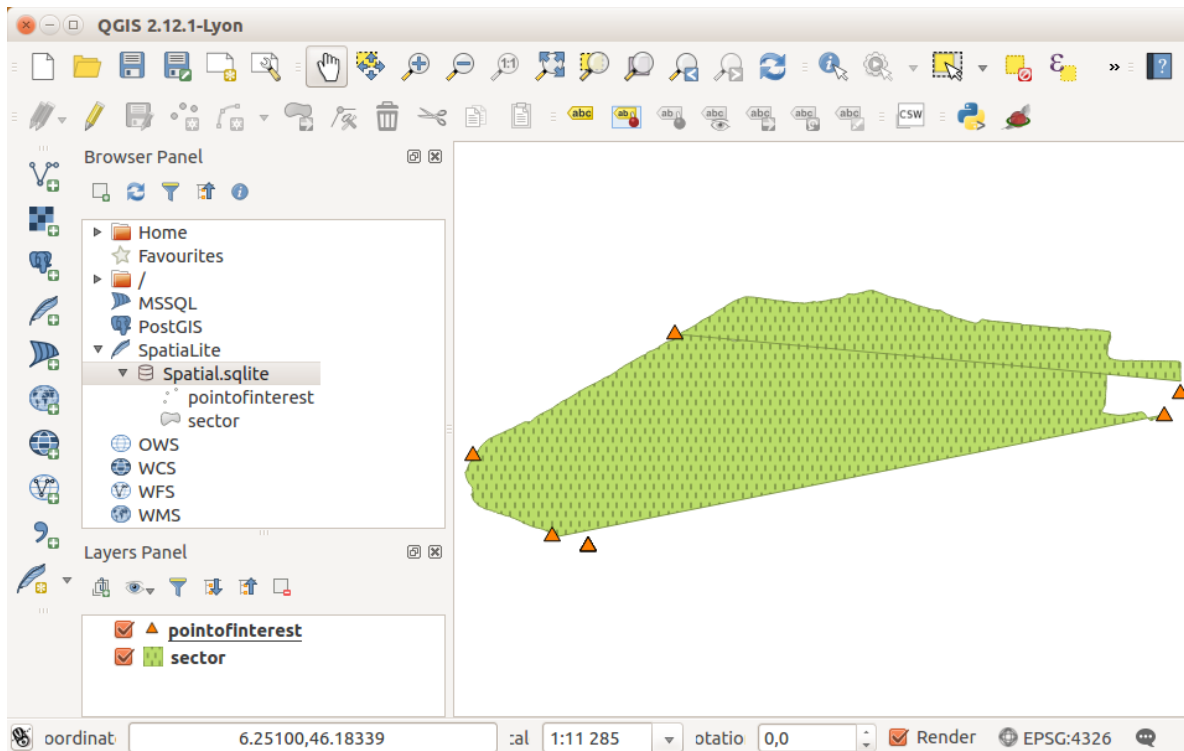


# Présentation

- Base de donnée Spatiale
- Open source (MPL 1.1, GPL 2, LGPL 2.1)
- Surcouche SQLite
- Originellement prévu pour PC et porté sur Android
- Embarqué
- Créer pour le projet geopaparazzi
- <https://github.com/geopaparazzi/libjsqlite-spatialite-android/wiki>

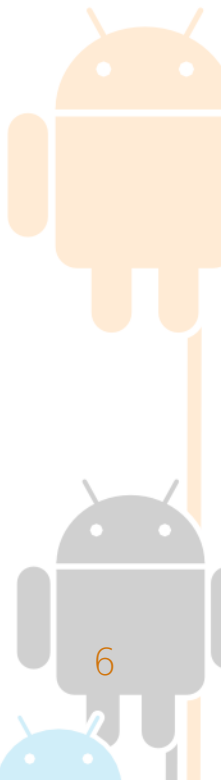


# Présentation



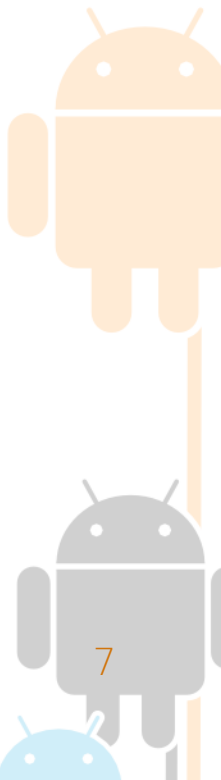
# Fonctionnalités

- Comme SQLite, db contenue dans un fichier
- Limites du File System
- Equi-fonctionnalité avec PostgreSQL + PostGIS
- Quantité de fonctions SQL : <http://www.gaia-gis.it/gaia-sins/spatialite-sql-4.3.0.html>
- Support les données géométriques conformément au standard OGC-SFS



# Présentation

- Portage de Spatialite sur Android
- C++ / JNI
- Compiler depuis les sources :
  - Nécessite d'être compilé avec le ndk (native development kit)
  - Il faut écrire les classes java associés
- Utilisé dans l'application geopaparazzi :  
<https://github.com/geopaparazzi/libsqlite-spatialite-android>



# Spatialite-Database-Driver

- Une bibliothèque Android Spatialite autonome
- Ecrit par Kristina Hager
- Repo git : <https://github.com/kristina-hager/Spatialite-Database-Driver>
- Extrait depuis l'application Geopaparazzi pour en créer une bibliothèque autonome





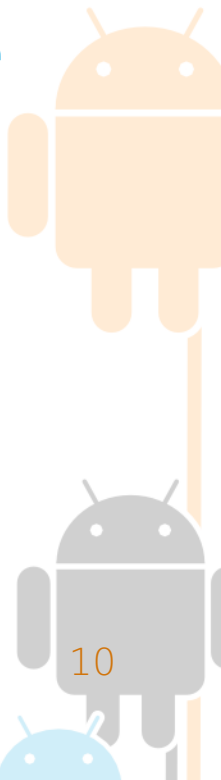
# IN01 – Séance 10

## Installation



# Installation

- Télécharger le repository git :  
<https://github.com/kristina-hager/Spatialite-Database-Driver>
- Placer la bibliothèque dans le même repertoire que le projet (bonne pratique pour les bibliothèques tiers)



# Installation

- Dans le projet cible :
  - ➔ Dans le module `app/build.gradle`, ajouter la dépendance vers la bibliothèque

```
dependencies {  
    compile project(  
        '...:Spatialite-Database-Driver:spatialite-db-driver')  
}
```

- ➔ Inclure la bibliothèque dans le setting du projet : `setting.gradle`

```
include ':app', '...:Spatialite-Database-Driver:spatialite-db-driver'
```



# Installation

- ➔ Dans le module app/build.gradle, ajouter la copie des bibliothèques natives dans la future apk

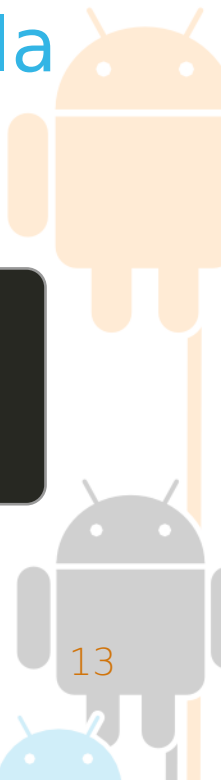
```
import com.android.build.gradle.tasks.PackageApplication

task copyNativeLibs(type: Copy) {
    from(new File(project(
        '...Spatialite-Database-Driver:spatialite-db-driver').projectDir,
        'src/main/java/jniLibs')) {
        include 'armeabi/libsqlite.so'
        include 'armeabi-v7a/libsqlite.so'
        include 'x86/libsqlite.so' // for emulator x86
    }
    into new File(buildDir, 'native-libs')
}
tasks.withType(JavaCompile) { compileTask -> compileTask.dependsOn copyNativeLibs }
clean.dependsOn 'cleanCopyNativeLibs'
tasks.withType(PackageApplication) { pkgTask ->
    // !!!! Nécessite la version 1.3.1 de gradle
    pkgTask.jniFolders = new HashSet<File>()
    pkgTask.jniFolders.add(new File(buildDir, 'native-libs'))
}
```

# Installation

- Attention, le code ci-dessus n'est pas compatible avec la version courante de gradle. Il faut revenir à une version antérieure
  - ➔ Dans le projet `app/build.gradle`, ajouter la dépendance vers la bibliothèque

```
Dependencies {  
    Classpath 'com.android.tools.build:gradle:1.3.1'  
}
```



# IN01 – Séance 10

Modélisation de base de donnée



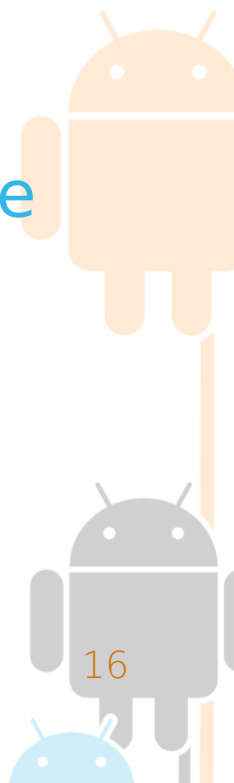
# Creation des tables

- Méthodologie et bonne pratique : toujours créer un MCD (conception) avant de commencer le developpement
- Traduire le MCD vers le MLD, format qui est compréhensible dans le SGBDR
- Créer les tables et leurs jointures
- Créer les indexes
- Le “+” spatial, ajouter les colonnes spatiales



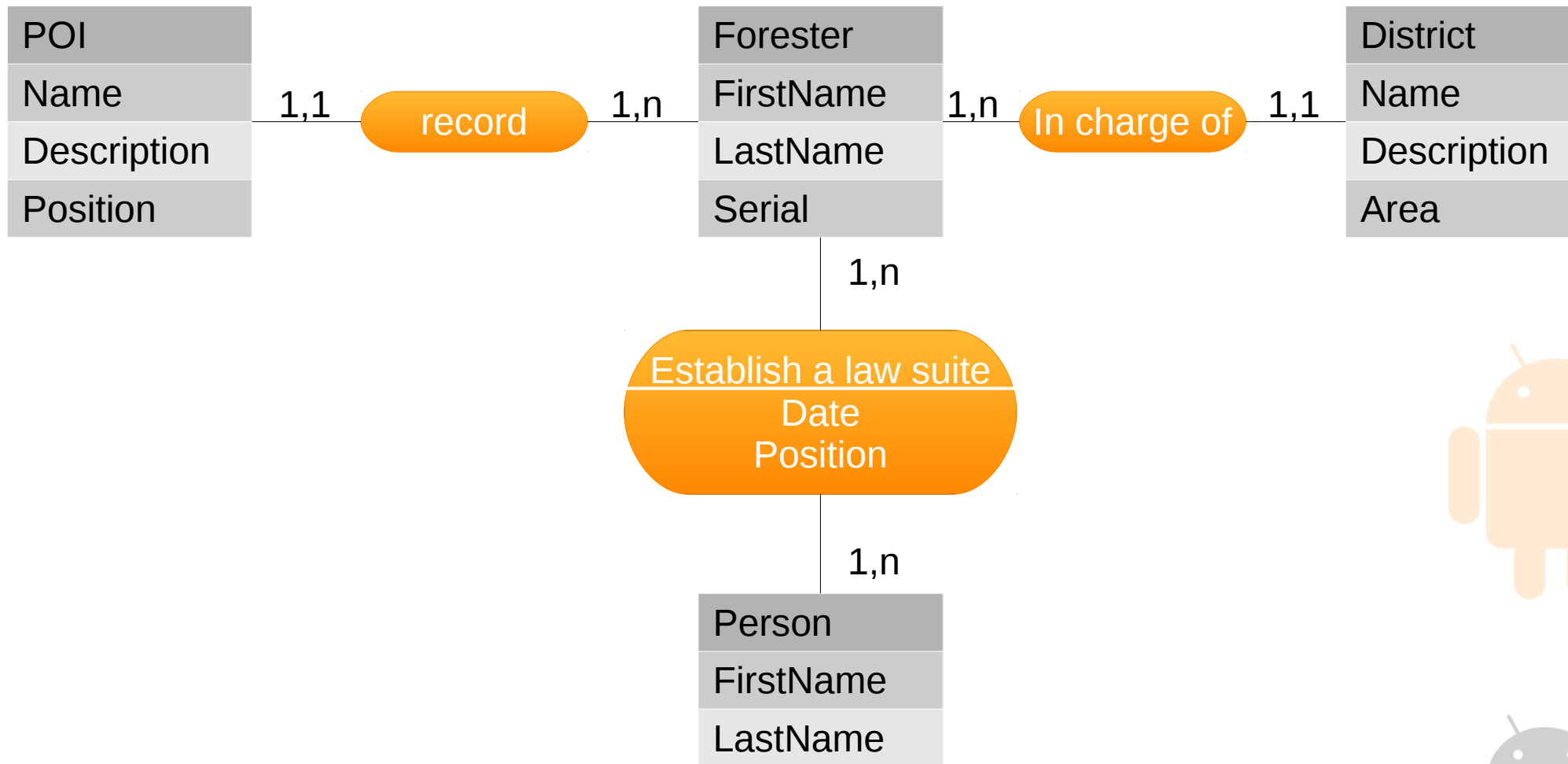
# Modélisation

- Divers méthodologies, objet (UML) ou entité-relation (Merise)
- Deux étapes :
  - ➔ Modélisation : création des entités et relations, regroupement des attributs
    - Grand principe ; éviter la redondance d'information
  - ➔ Normalisation : convertir le model vers la machine
    - Les jointures n / n deviennent une table avec 2 Clés étrangères
    - Les classes sur-type sous-types sont normalisées

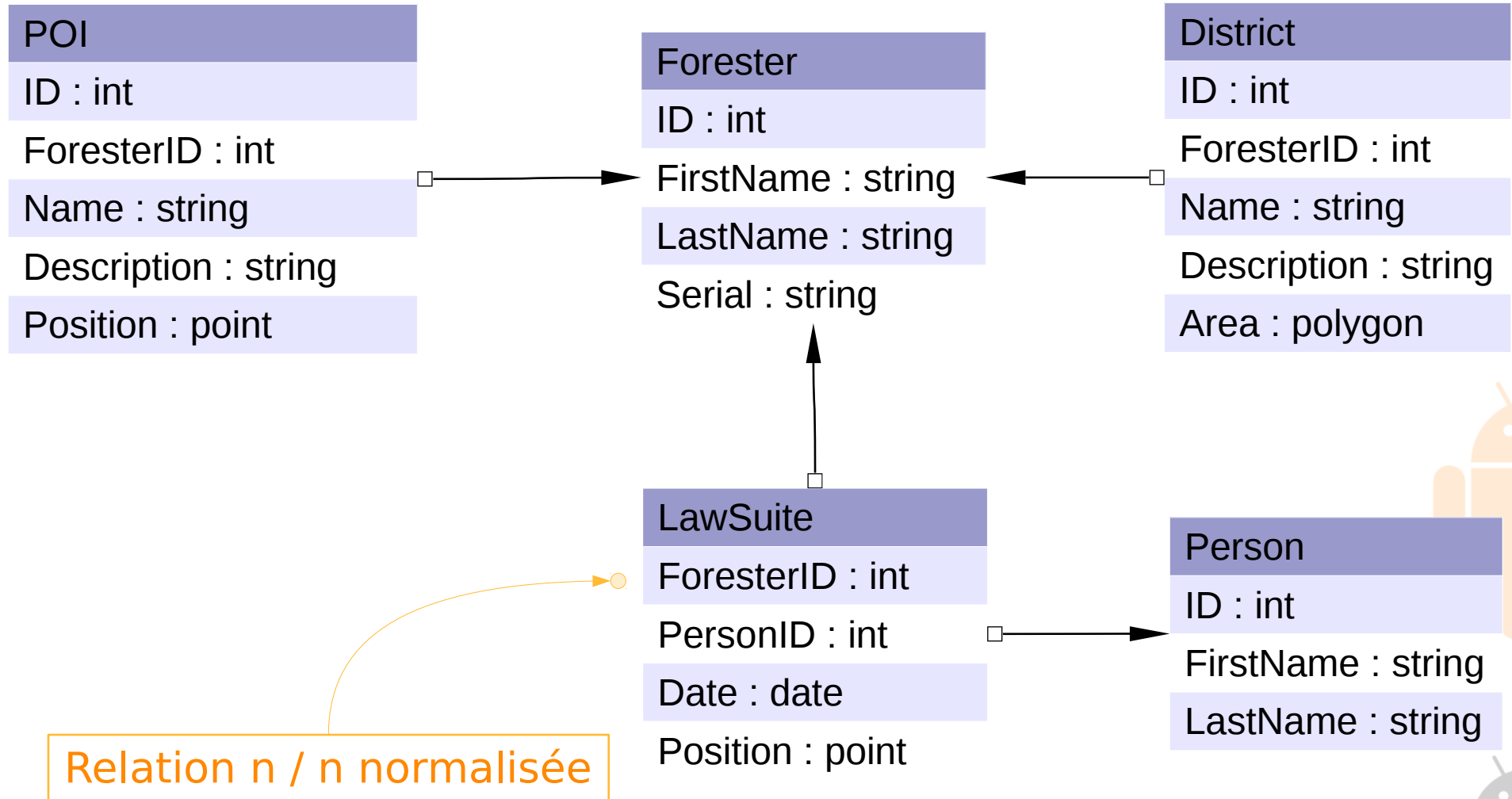




# Exemple de MCD



# MLD



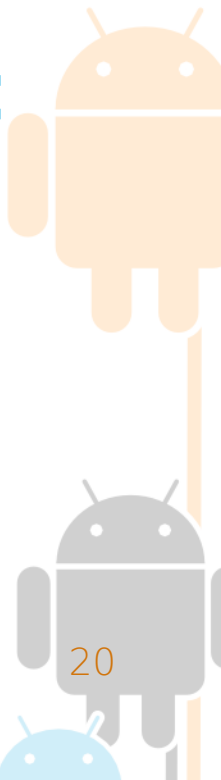
# IN01 – Séance 10

Création de base de donnée



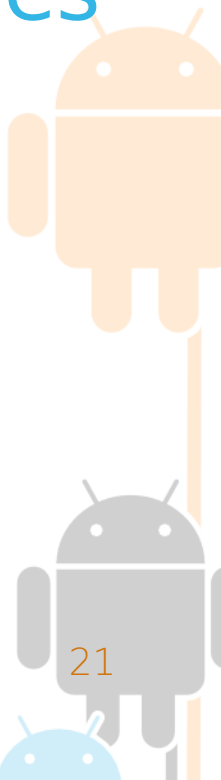
# Spatial MetaData Tables

- A la création de la base de donnée, spatialite requière des tables systèmes particulières
- Appellées MetaData Tables
- Pour les créer, il faut lancer la fonction :  
`SELECT InitSpatialMetaData();`

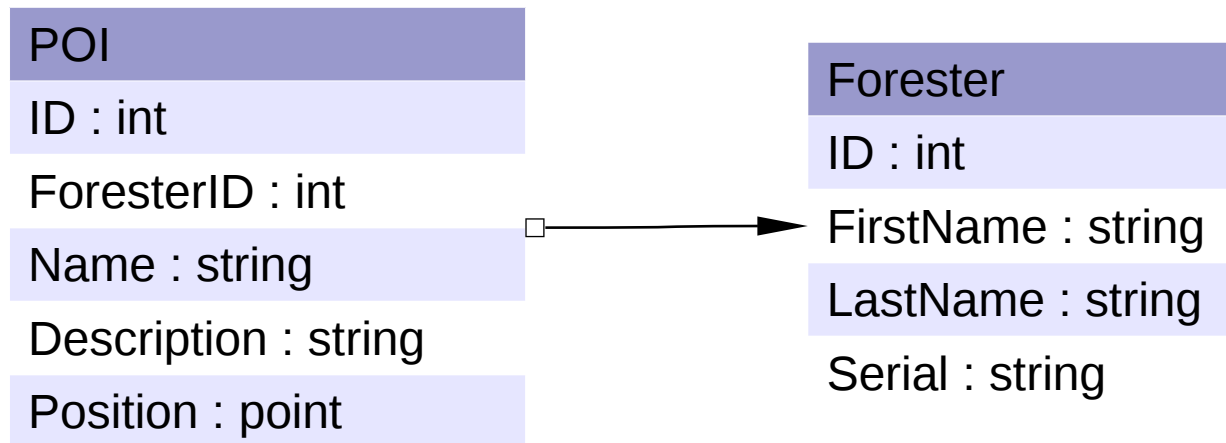


# Spatial MetaData Tables

- Quelques exemples de tables MetaData
- `spatial_ref_sys` : contient les données EPSG (European Petroleum Survey Group)
- `geometry_columns` : recense les colonnes géométrique de la base de donnée



# Exemple de MLD



# Création de table

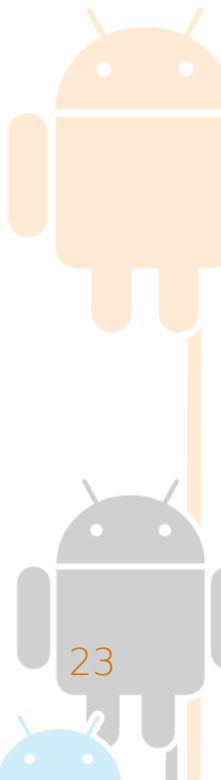
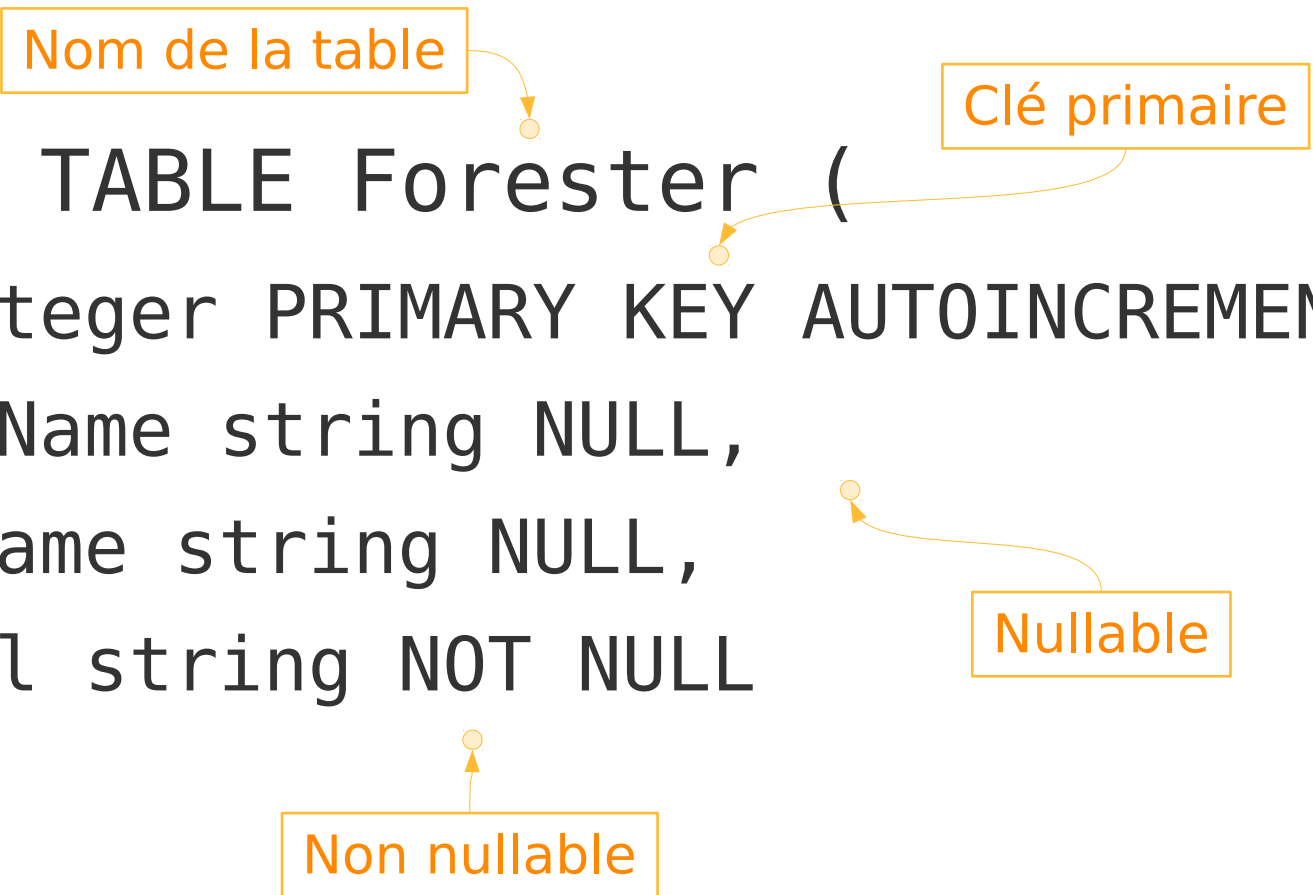
■ `CREATE TABLE Forester (`  
    `ID integer PRIMARY KEY AUTOINCREMENT,`  
    `FirstName string NULL,`  
    `LastName string NULL,`  
    `Serial string NOT NULL`  
`);`

Nom de la table



Clé primaire

Nullable

Non nullable



# Création de table

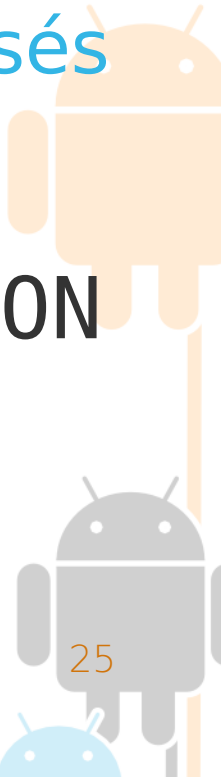
- ```
CREATE TABLE PointOfInterest (  
    id integer PRIMARY KEY AUTOINCREMENT,  
    foresterID integer NOT NULL,  Clé étrangère  
    name string NOT NULL,  
    description string,  
    CONSTRAINT FK_poi_forester  
        FOREIGN KEY (foresterID)  Contrainte clé étrangère  
        REFERENCE forester (id)  
);
```





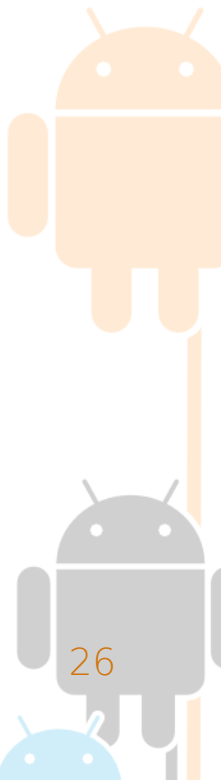
# Création d'index

- Les indexes servent à optimiser les performances de la base de donnée
  - ➔ Sur les jointure : il faut un index sur chaque clé étrangère
  - ➔ Sur les critères de recherches souvent utilisés
- `CREATE INDEX IDX_poi_forester_id ON PointOfInterest (forester_id);`



# Création d'une table géométrique

- Comme Spatialite est une surcouche à SQLite, il n'existe pas de grammaire native pour créer des données Géométriques
- Il est donc nécessaire de procéder en deux étapes :
  - ➔ La création de la table en SQL
  - ➔ L'ajout de la (des) colonne(s) géométrique(s) :  
fonction `AddGeometryColumn`



# Création de la colonne géométrique

- **SELECT**  
`AddGeometryColumn( 'PointOfInterest'`  
`, 'position', 4326, 'POINT', 'XY',`  
`0);`

Géométrie Nullable

SRID

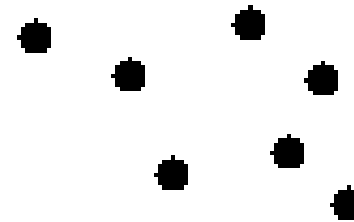
Type de géométrie

Nb dimensions

# Geometry : POINT, MULTIPOINT

● [x,y]

POINT

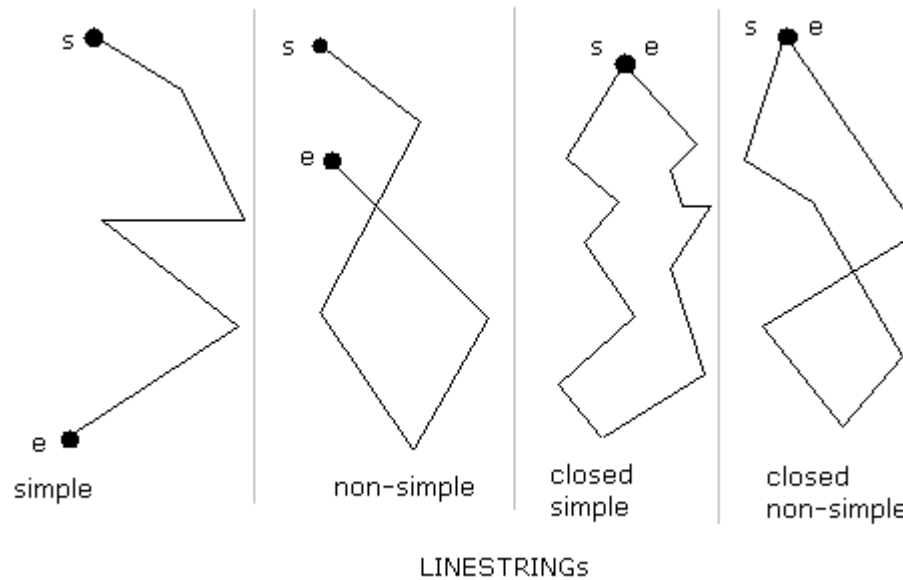


MULTIPOINT

- POINT(123.45 543.21)
- MULTIPOINT(1234.56 6543.21, 1 2, 3 4, 65.21 124.78)

Succession de points

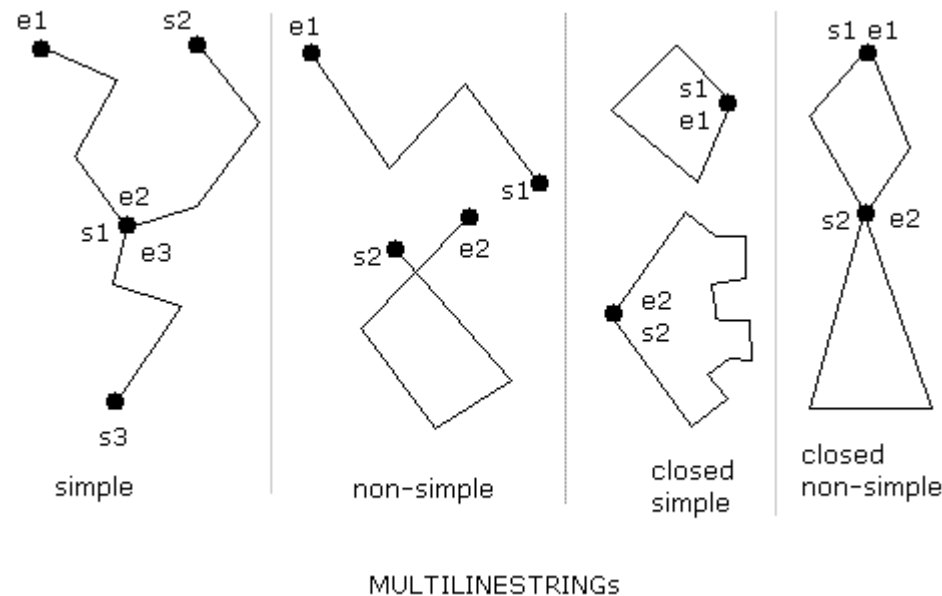
# Geometry : LINESTRING



- `LINESTRING(100.0 200.0, 201.5 102.5, 1234.56 123.89)`

Succession de points

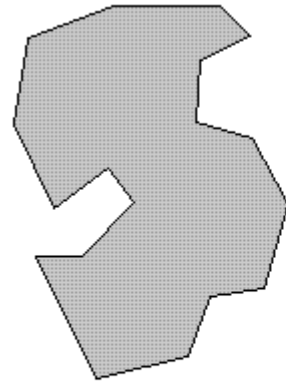
# Geometry : MULTILINESTRING



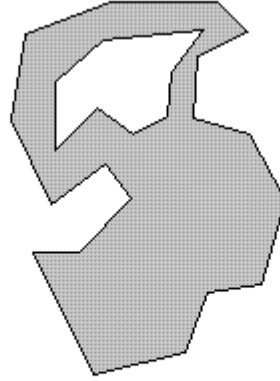
- MULTILINESTRING((1 2, 3 4), (5 6, 7 8, 9 10), (11 12, 13 14))

Succession de linestrings

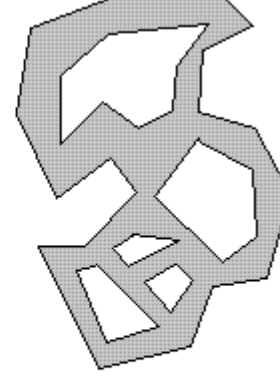
# Geometry : POLYGON



exterior ring  
no interior rings



exterior ring  
1 interior ring



exterior ring  
5 interior rings

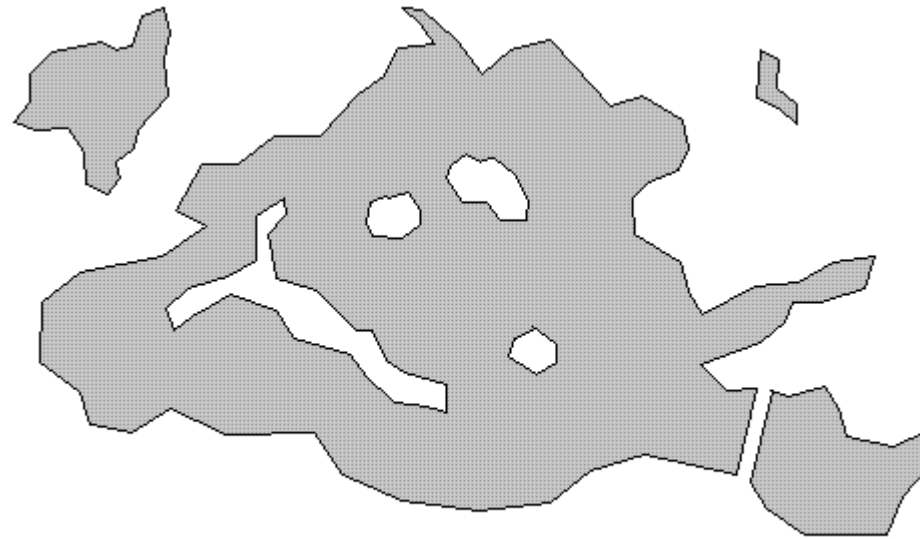
POLYGONS

Bordure extérieure

- `POLYGON( (10 10, 20 10, 20 20, 10 20, 10 10), (13 13, 17 13, 17 17, 13 17, 13 13) )`

Bordure intérieure

# Geometry : MULTIPOLYGON



MULTIPOLYGON

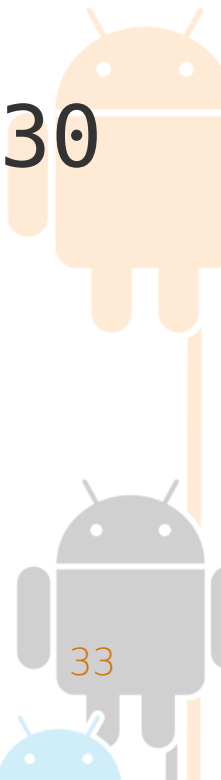
- `MULTIPOLYGON(((0 0,10 20,30 40,0 0),(1 1,2 2,3 3,1 1)),`
- `((100 100,110 110,120 120,100 100)))`

Succession de polygones



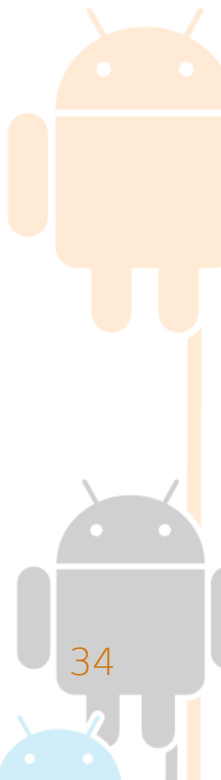
# Geometry : GEOMETRYCOLLECTION

- Une succession d'objets géométriques
- Non standard et peu reconnu
- `GEOMETRYCOLLECTION(POINT(1 1),  
LINESTRING(4 5, 6 7, 8 9), POINT(30  
30))`



# Dimensions

- Quatre systèmes de dimensions possible
  - XY : Coordonnées 2D
  - XYM : Coordonnées 2D + Mesure
  - XYZ : Coordonnées 3D
  - XYZM : Coordonnées 3D + Mesure



# Dimensions

- Dont on dérive les géométries

| XY              | XYM               | XYZ               | XYZM               |
|-----------------|-------------------|-------------------|--------------------|
| POINT           | POINT M           | POINT Z           | POINT ZM           |
| MULTIPOINT      | MULTIPOINT M      | MULTIPOINT Z      | MULTIPOINT ZM      |
| LINESTRING      | LINESTRING M      | LINESTRING Z      | LINESTRING ZM      |
| MULTILINESTRING | MULTILINESTRING M | MULTILINESTRING Z | MULTILINESTRING ZM |
| POLYGON         | POLYGON M         | POLYGON Z         | POLYGON ZM         |
| MULTIPOLYGON    | MULTIPOLYGON M    | MULTIPOLYGON Z    | MULTIPOLYGON ZM    |

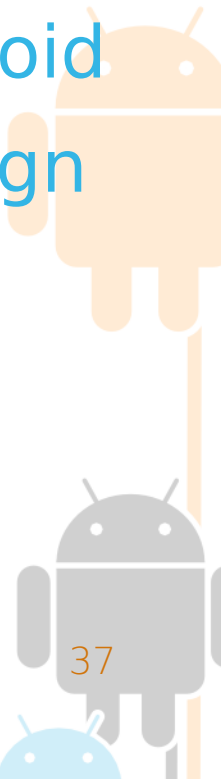
# IN01 – Séance 10

## Spatialite Helper



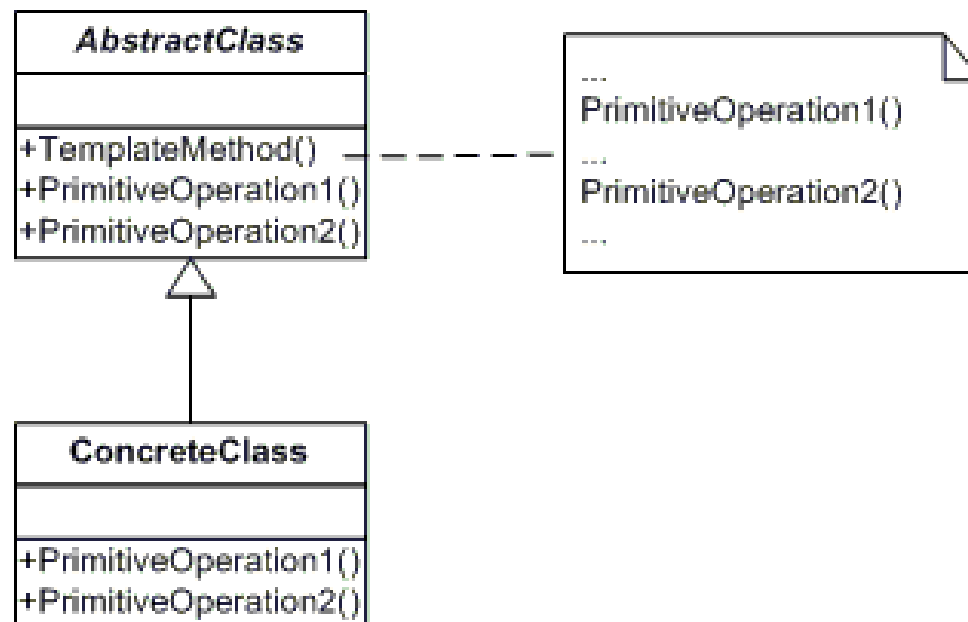
# Spatialite Helper

- Développé pour le TP
  - ➔ Basé sur SqliteOpenHelper
  - ➔ Prend en charge les MetaData Tables
  - ➔ Facilite la création et la gestion des mises à jour dans le contexte d'une application Android
  - ➔ Permet de manipuler et comprendre un design pattern : Template Method
    - Propre à l'architecture en couche

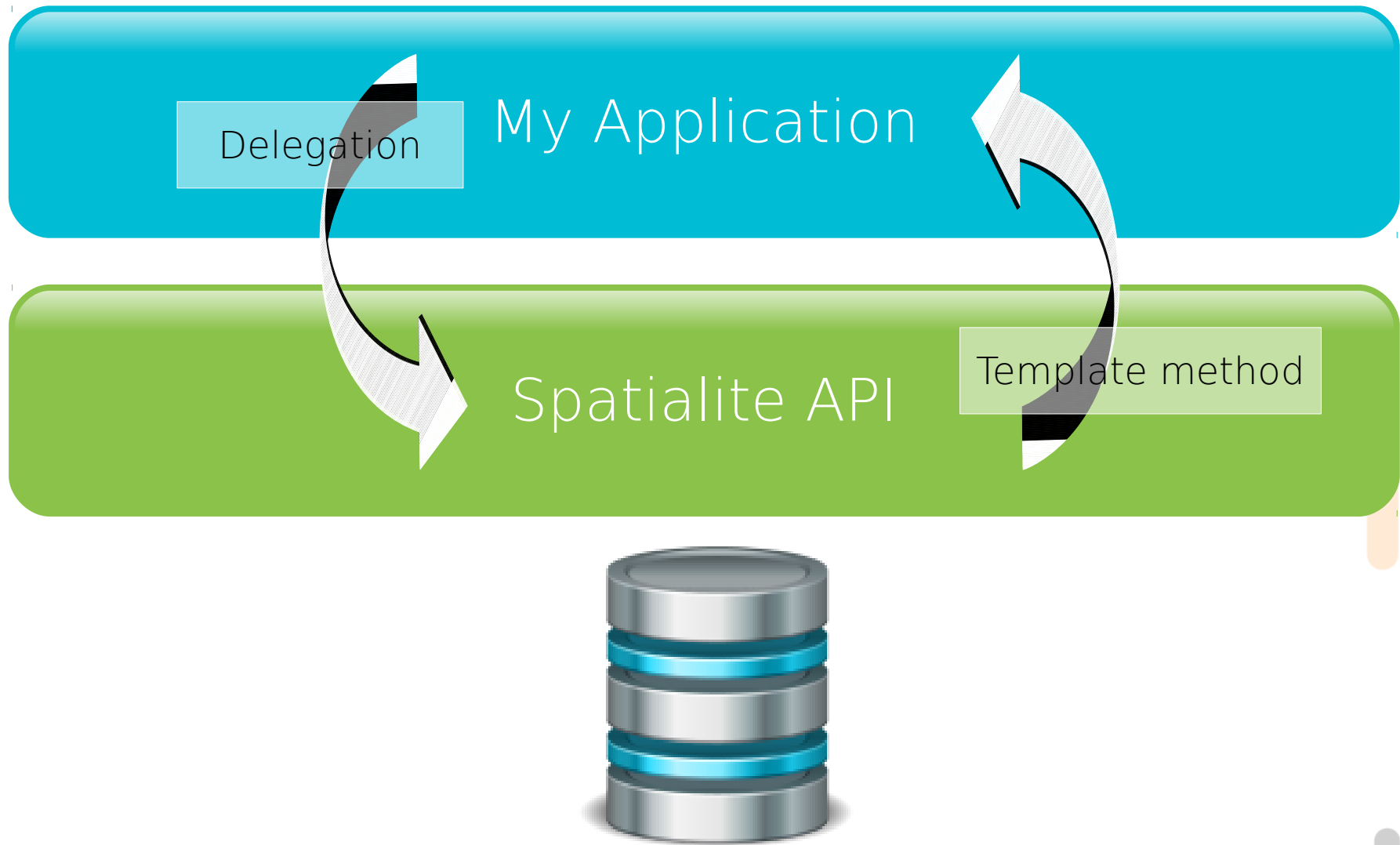


# Template Method

- Lorsqu'un méthode concrète d'une classe abstraite fait appel à des méthodes abstraites de celle ci

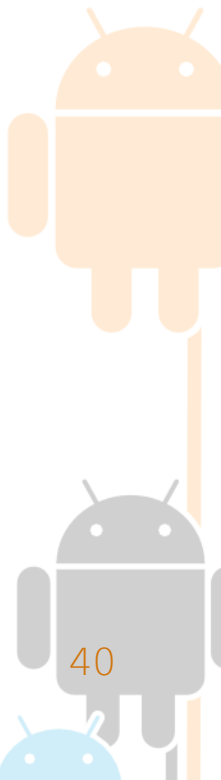


# Architecture en couche



# SpatialiteOpenHelper

- Créer une classe qui hérite de `SpatialOpenHelper`
- Surcharger la méthode `onCreate` ; en charge de créer le schema de la base
- Surcharger la méthode `onUpgrade` ; en charge de gérer le changement de version de l'application
- Utiliser la méthode `super.exec`





# SpatialiteOpenHelper

```
public class SDBHelper extends SpatialiteOpenHelper {  
  
    @Override  
    public void onCreate(Database db) throws SQLException {  
  
        // Création du schema  
    }  
  
    @Override  
    public void onUpgrade(Database db, int oldVersion, int newVersion)  
        throws Exception {  
  
        // Upgrade  
    }  
}
```

Lancement à l'installation  
Schema complet et à jour

Lorsque l'application est mise à jour par l'utilisateur

# Création du schema

```
@Override
public void onCreate(Database db) throws SQLException {
    // Création du schema
    super.exec("create table Forester (" +
        "id integer PRIMARY KEY AUTOINCREMENT, " +
        "firstName string NOT NULL, " +
        "lastName string NOT NULL, " +
        "serial string NULL);");

    super.exec("create table PointOfInterest (" +
        "id integer PRIMARY KEY AUTOINCREMENT, " +
        "foresterID integer NOT NULL, " +
        "name string NOT NULL, " +
        "description string, " +
        "CONSTRAINT FK_poi_forester " +
        "FOREIGN KEY (foresterID) " +
        "REFERENCE forester (id) );");

    super.exec("CREATE INDEX IDX_poi_forester " +
        "ON PointOfInterest (forester_id);");

    super.exec("SELECT AddGeometryColumn(" +
        "'PointOfInterest', 'position', 4326, 'POINT', 'XY', 0);");
}
```

# Gestion de la version

- Le numéro de version de la base de donnée est passé en paramètre au super constructeur
- La super classe abstraite se chargera de gérer la mise à jour

```
public MySpatialiteHelper(Context context) throws .... {  
    super(context, "Spatial.sqlite", 4);  
}
```

# Gestion des mises à jours

- Problématique : que deviens le schema si l'application est mise à jours par l'utilisateur

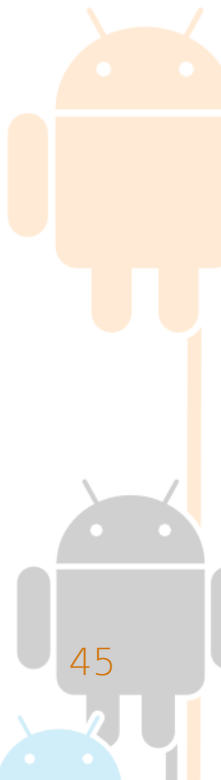
```
@Override
public void onUpgrade(Database db, int oldVersion, int newVersion) {
    switch(oldVersion) {
        case 1:
            // mise à jour de 1 -> 2
            // pas de break
        case 2:
            // mise à jour de 2 -> 3
            // pas de break
        case 3:
            // mise à jour de 3 -> 4
            break;
        default:
            throw new IllegalStateException(
                "onUpgrade() with unknown oldVersion" + oldVersion);
    }
}
```

Gère les changements  
étape par étape

Exécution en cascade

# Mise à jour de l'application

- Si le schema est modifié, il faut ajouter les requête de création à deux endroits :
  - ➔ onCreate : pour que les nouveaux utilisateurs aient le nouveau schéma directement
  - ➔ onUpdate : ajouter un case au switch et déplacer le break, pour que les utilisateurs possédant une ancienne version aient leur schema mis à jour



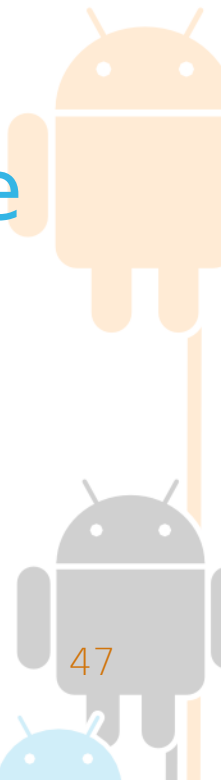
# IN01 – Séance 10

## Requêtes



# WKT / WKB

- WKT : Well known Text, exploitable par un être humain
- WKB : Well Known Binary, exploitable par la machine à destination d'import, export et échanges
- Contre un format de stockage Spatialite  
BLOB Geometry
- Il faut convertir !



# Fonctions de conversions

- ST\_GeomFromText : converti WKT vers BLOB Geometry

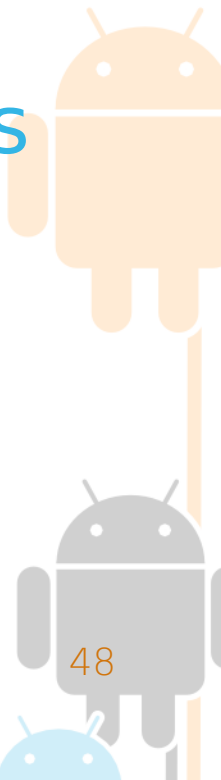
→ `SELECT ST_GeomFromText( 'POINT(1.2345 2.3456) ' );`

- `GeomObject`

- ST\_AsText : converti BLOB Geometry vers WKT

→ `SELECT ST_AsText(x'0001FFFFFFFF ... ');`

- `POINT(1.2345 2.3456)`





# Fonctions de conversions

- ST\_GeomFromWKB : converti WKB vers BLOB Geometry

→ ST\_GeomFromWKB(x'010100000 ... ');

- GeomObject

- ST\_AsBinary : converti BLOB Geometry vers WKB

Hexadecimal

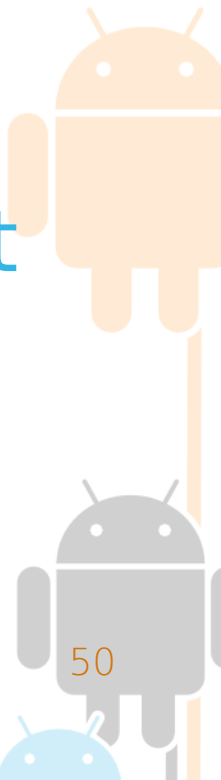
→ SELECT  
HEX(ST\_AsBinary(x'010100000 ... '));

- 01010000008D976E1283C0F33F16FBCBEEC9C30240



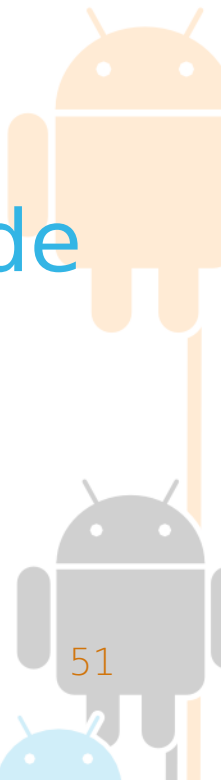
# Fonctions utiles

- `ST_GeometryType` : renvoie le type de géométrie
  - ➔ `SELECT`  
`ST_GeometryType(ST_GeomFromText('POINT M(1.2345 2.3456)'));`
    - `POINT M`
- Attention, les deux syntaxes `POINT M` et `POINTM` sont valides



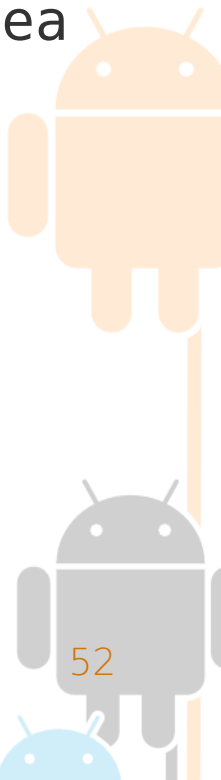
# Fonctions utiles

- `ST_Srid` : renvoie le SRID (EPSG) de la valeur géométrique
  - ➔ `SELECT`  
`ST_Srid(ST_GeomFromText( 'POINT(1.2345 2.3456) ', 4326));`
    - 4326
- Attention, à toujours spécifier les SRID de la géométrie !



# Autres Fonctions

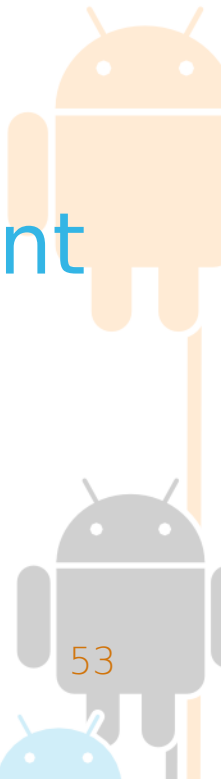
- **Informations** : `spatialite_version`, `spatialite_target_cpu`
- **Conversion** : `CastToMultiLineString`, `CastToXYZM`, `CastToXY`
- **Conversion géographique** : `LongLatToDMS`, `LongitudeFromDMS`, `LatitudeFromDMS`
- **Unités** : `CvtToKm`, `CvtFromKm`, `CvtToUsIn`, `CvtFromUsIn`
- **Calculs géométriques** : `ST_Perimeter`, `ST_Centroid`, `ST_Area`
- **Relation géométriques** : `ST_Overlaps`, `ST_Intersects`, `ST_Contains`
- **Avancé** : `eval`, `AsSVG`, `AsKml`, `AsGeoJSON`
- **Référence** : <http://www.gaia-gis.it/gaia-sins/spatialite-sql-4.3.0.html>



# Exécuter une requêtes : code

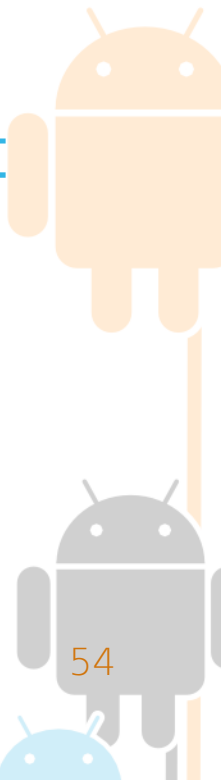
```
helper.exec(
    "insert into " + MySpatialiteHelper.TABLE_SECTOR +
    "(" + MySpatialiteHelper.COLUMN_NAME + ", "
    + MySpatialiteHelper.COLUMN_COORDINATE + ")" +
    " values ('" + name.getText() + "', '"
    + comment.getText() + "', "
    + shape.toSpatialiteQuery(MySpatialiteHelper.GPS_SRID) + "');" );
```

- S'exécute grâce au helper
- Il faut gérer les exceptions le cas échéant



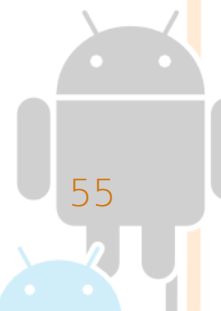
# jsqlite.stmt

- Pour lire le résultat d'une requête
- Appellé ResultSet ou Statement
- Il faut utiliser la méthode `Database.Prepare(sql)` qui renvoie un Stmt
- Itérer avec la méthode `stmt.step()`
- Récupérer les valeurs typés de chaque élément avec `stmt.column_string(columnId)`, `column_int` etc...
- Gérer les erreurs éventuelles



# Stmt : code

```
stmt = database.prepare("Select name, description, " +  
    "ST_asText(coordinate) as coord from PointOfInterest");  
  
while (stmt.step()) {  
    String name = stmt.column_string(0);  
    String comment = stmt.column_string(1);  
    String coordStr = stmt.column_string(2);  
  
    Point coord = Point.unMarshall(new StringBuilder(coordStr));  
    mapFragment.addMarker(coord, name, comment);  
}
```



# Exemple de jointure

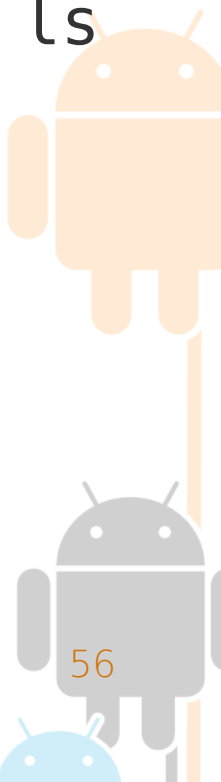
- Tous les secteurs d'un garde forestier

- ➔ `SELECT s.Name, AsText(s.Area) FROM Sector s  
INNER JOIN forester f ON f.id = s.foresterID`

- Les contraventions bornés dans un temps donné

- ➔ `SELECT ls.Date, f.FirstName FROM LawSuite ls  
INNER JOIN forester f ON f.id =  
ls.foresterID`

- `WHERE ls.date BETWEEN '2015-01-01' AND  
'2015-02-01'`





# Spatiale

- Jointure :

- `SELECT s.name, poi.name from Sector s, PointOfInterest poi where ST_Contains(s.area, poi.position)`

- Ou (Sql99) :

- `SELECT s.name, poi.name from Sector s INNER JOIN PointOfInterest poi ON ST_Contains(s.area, poi.position)`

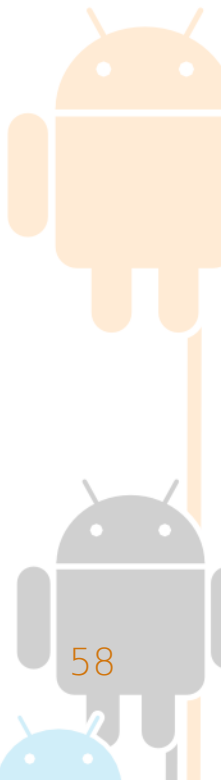
- Area

- `SELECT *, ST_Area(area) as 'area' from Sector`



# Spatiale

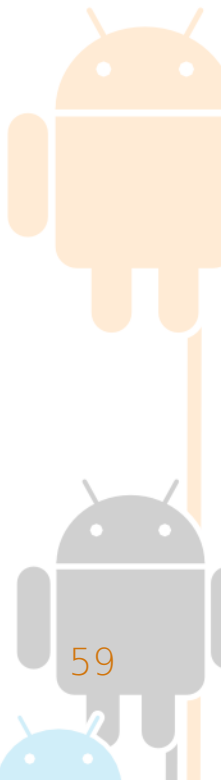
- Le plus petit secteur
  - ➔ `SELECT MIN(St_Area(area)) from Sector`
- La plus grande distance entre deux points
  - ➔ `SELECT MAX(St_Distance(poi1.position, poi2.position)) from PointOfInterest poi1, PointOfInterest poi2`



# Spatiale

- Multi jointure et jointure spatiale :

```
→ SELECT * FROM LawSuite ls
    INNER JOIN Sector s ON St_Contains(s.area,
    poi.position)
    INNER JOIN Forester f ON f.id =
    s.foresterId AND f.id = ls.ForesterId
WHERE s.name = 'Fontainbleau'
```



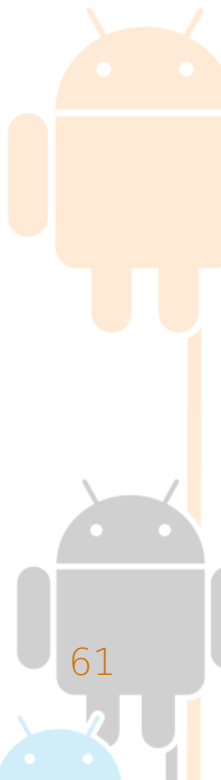
# IN01 – Séance 10

## Objets géométriques



# Présentation

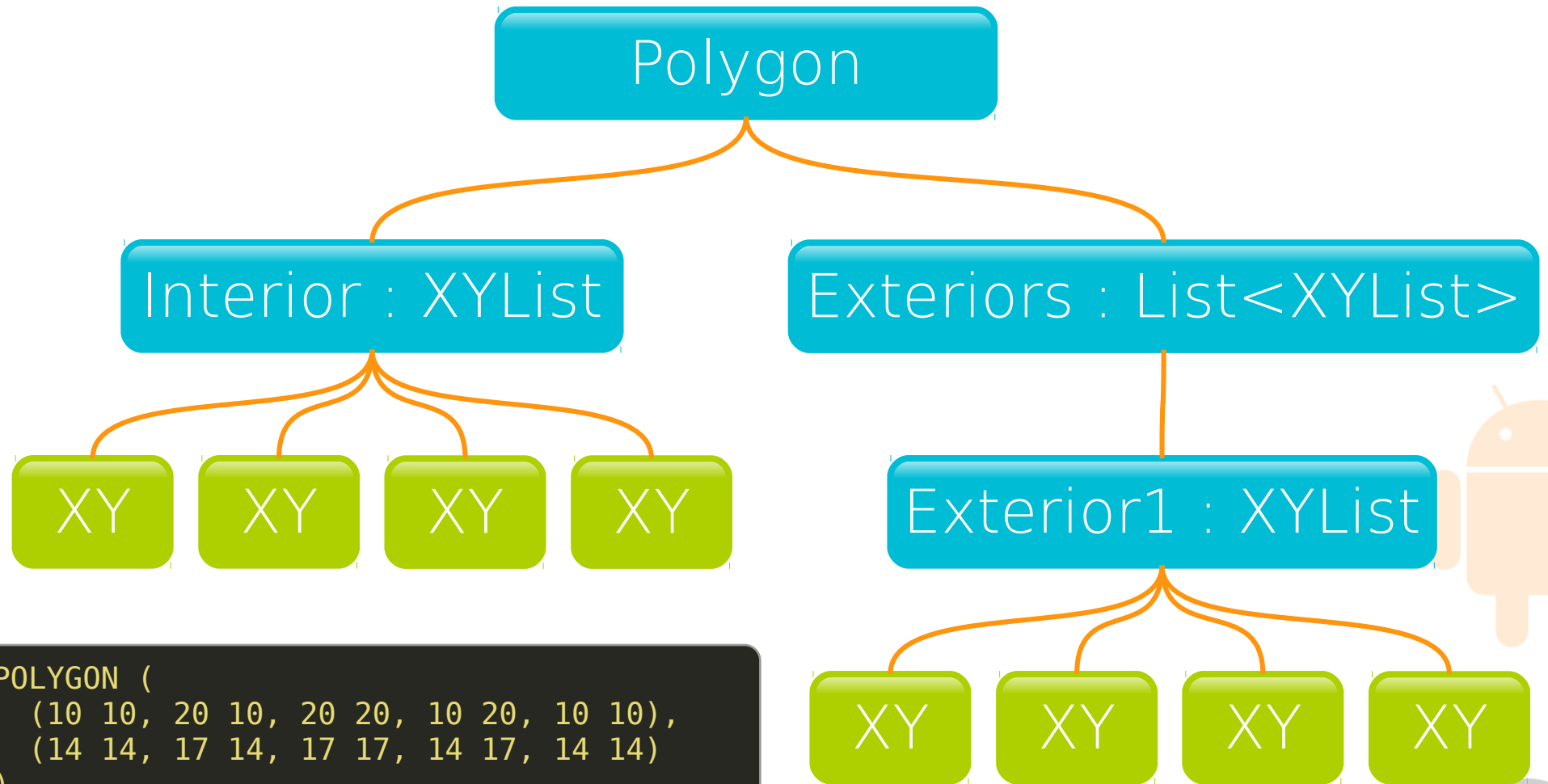
- Librairie conçue pour le TP
- Conversion WKT → Objet (unmarshall) et Objet → WKT (marshall)
- Idée de base, manipuler des objets plutôt que des chaînes de caractères
  - Eviter les erreurs
  - Bénéficier du typage fort du langage Java
  - Facile à maintenir
  - Pas de code tiers dans les couches hautes
  - Optimiser la concatenation des chaînes de caractères
- Bonne pratique (requêtes Sql)



# Exemple d'utilisation

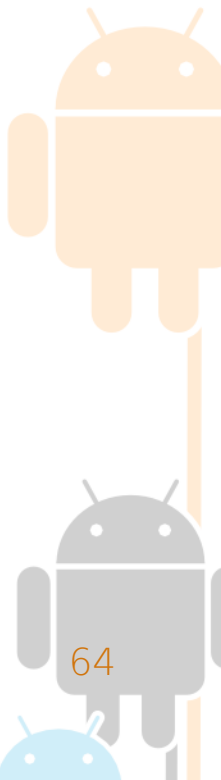
```
public void testMarshall() throws Exception {  
    Polygon polygon = new Polygon();  
    polygon.addCoordinate(new XY(10, 10));  
    polygon.addCoordinate(new XY(20, 10));  
    polygon.addCoordinate(new XY(20, 20));  
    polygon.addCoordinate(new XY(10, 20));  
  
    XYList interior = new XYList(true);  
    interior.add(new XY(14, 14));  
    interior.add(new XY(17, 14));  
    interior.add(new XY(17, 17));  
    interior.add(new XY(14, 17));  
    polygon.addInterior(interior);  
  
    assertEquals("POLYGON ((10 10, 20 10, 20 20, 10 20, 10 10),  
(14 14, 17 14, 17 17, 14 17, 14 14))", polygon.toString());  
}
```

# Parcours en profondeur



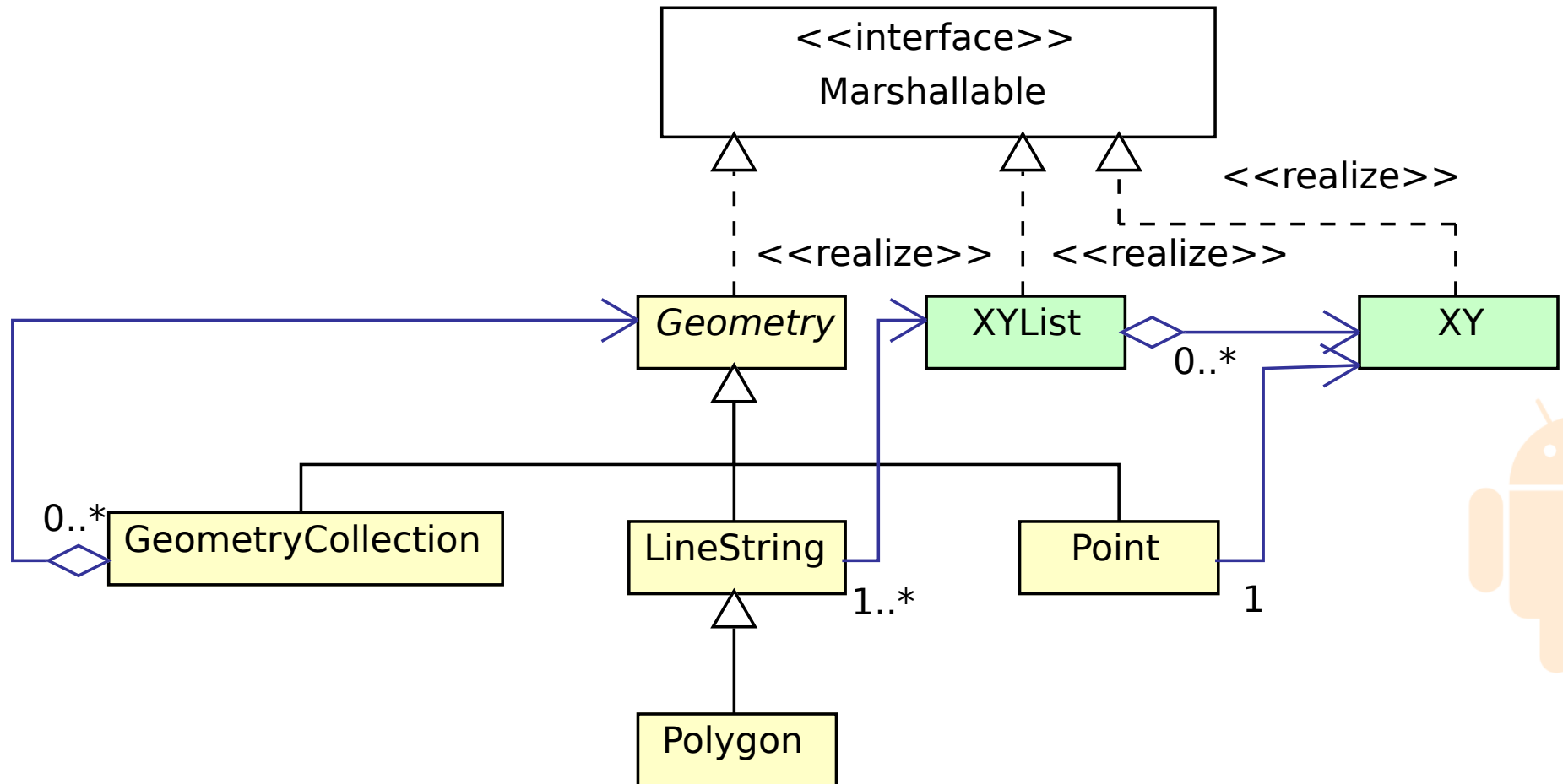
# Principes

- POO :
  - ➔ Hierarchie de type et Polymorphisme
  - ➔ Hierarchie d'objets ; Design pattern Composite
  - ➔ Une interface commune marshallable
- Algorithmie :
  - ➔ Un parcours en profondeur
  - ➔ Un Recursive Descent Parser





# UML



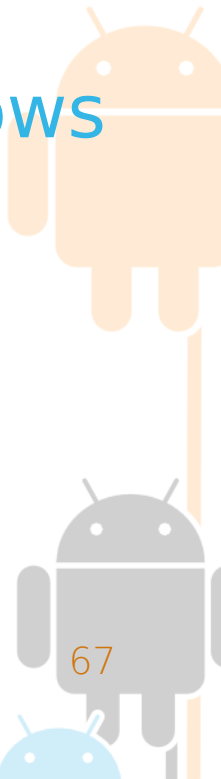
# IN01 – Séance 10

Pour aller plus loin



# ORM

- Object Relational Mapping
  - ➔ Pour Sqlite, il existe un ORM appelé ORM-Lite
  - ➔ Il n'existe pas, à ce jour, d'extension spatiale
- Et les raster ?
  - ➔ Il existe librasterlite2 pour la version Windows
  - ➔ Il faudrait le porter sur Android
    - Compiler les sources C++
    - Créer le mapping JNI



# WebServices

- Spatialite est une base de données embarquée et mono application / utilisateur
- Idée de mutualiser les données
- Synchronisation avec une base de donnée centralisée via WebServices (cf. Chapitre 09)
- Exemple : PostGIS / Jax WS / Glassfish

# Fin

- Merci de votre attention
- Des questions ?

